CMPSC-265
Data Structures and Algorithms

Zaihan Yang
zyang13@suffolk.edu

Department of Math and Computer Science
Suffolk University

Fall 2019

Notice

- Midterm_Exam2 grades posted.
- HW11 posted.

Review

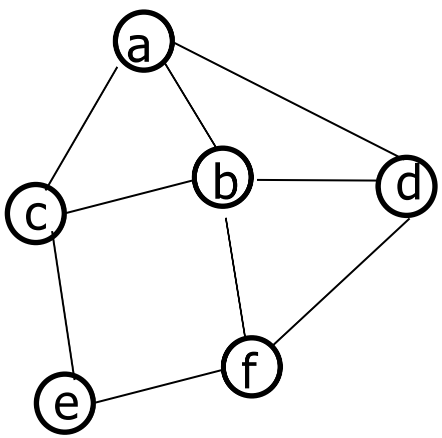
- Graph and its representation
- Graph traversal algorithms:
  - Depth-first-search (DFS)
  - Breadth-first-search (BFS)

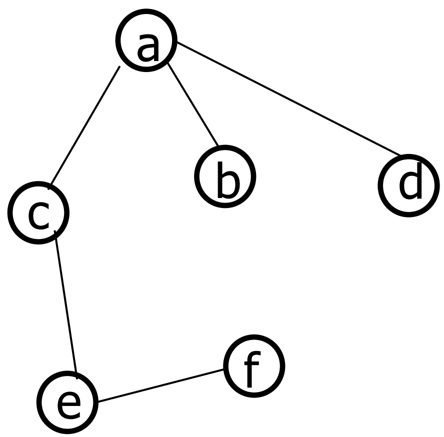
Learning Topics

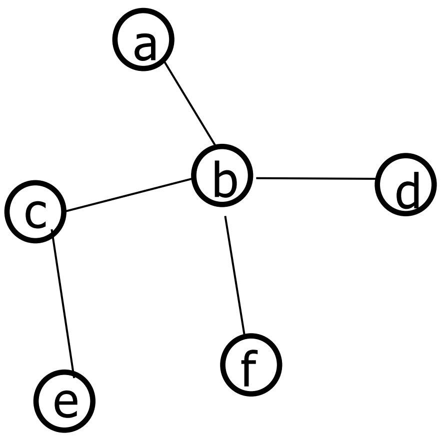
- Minimum Spanning Tree
- How to find minimum Spanning Tree on an undirected non-weighted graph
- How to find Minimum Spanning Tree on an undirected weighted graph:
  - Prim’s Algorithm
  - Krustal’s Algorithm

Minimum Spanning Tree

- Spanning Tree: A connected subgraph with no cycles including all N vertices
- Minimum Spanning Tree(MST)on a non-weighted graph is a spanning tree with minimum number of edges.
- A minimum spanning tree (MST) of an edge-weighted graph is a spanning tree whose weight (the sum of the weights of all its edges) is no larger than the weight of any other spanning tree.

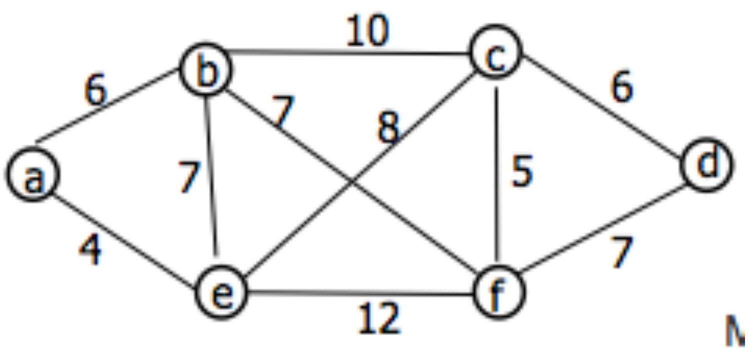


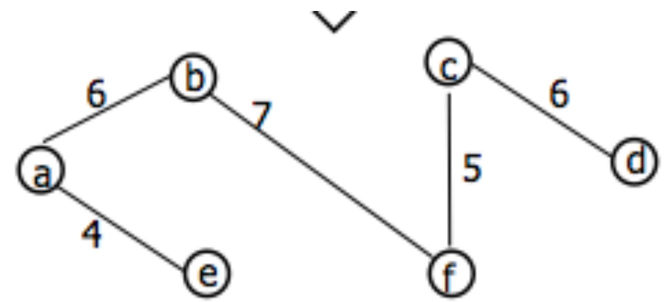




Minimum Spanning Tree

- MST on weighted graph





- MST has many applications:

application	vertex	edge
circuit	component	wire
airline	airport	flight route
power distribution	power plant	transmission lines
image analysis	feature	proximity relationship

Typical MST applications

Minimum Spanning Tree

- The graph is connected for an MST to exist. If the graph is not connected, we can find the MST on each connected components, and thus from a minimum spanning forest.
- How many edges are in MST of a graph of N vertices?
  - N-1 edges is enough to connect N vertices
- Is MST unique?
  - No
- How to find/create MST on a non-weigtted graph?
  - Similar to search methods, but record edges as you search/ traverse
  - Depending on the starting vertex, we might get a different MST

Minimum Spanning Tree

- Finding MST using DFS:

```
public void mst(){
   nodes[0].visited=true;
   nodes[0].display();
   theStack.push(0);
   while(!theStack.isEmpty()){
      int current=theStack.peek();
      int v=getAdjUnvisitedNode(current);
      if (v ==-1) // no such node
         theStack.pop();
      else {
         nodes[v].visited=true;
         theStack.push(v);
         //display edge
         nodes[current].display(); //from
         nodes[v].display(); //to
         System.out.print(" ");
}
   }
   // stack is empty, reset the flags
   for(int i=0; i<N;i++)
      nodes[i].visited=false;
}
```

MST on Weighted Graphs

- Edges have weights (numeric values).
  - Can represent cost/distance or value/profit.

- In a weighted graph, we find a spanning tree with minimum total (sum) weight. Example: installing cable tv links among cities, and other applications:

- There are two well-known algorithms for this purpose:
  - Prim’s algorithm
  - Kruskal’s algorithm
- Both algorithms use a *greedy* approach which is based on making locally optimal decisions at every step.

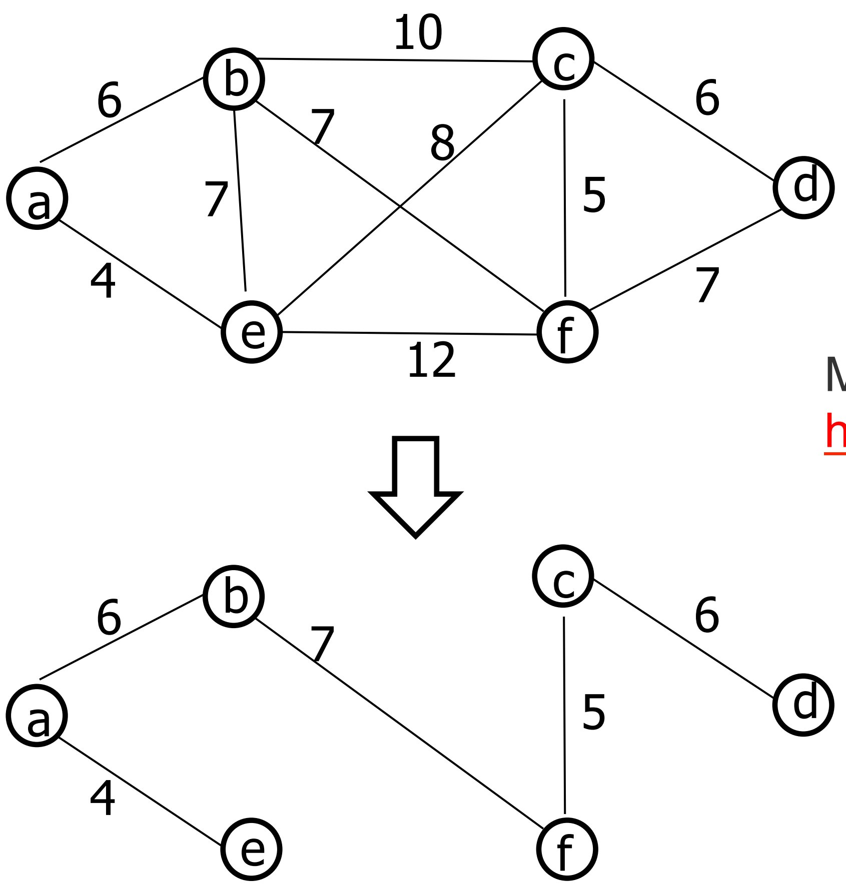
Prim’s Algorithm for finding MST

- Start with an arbitrary vertex as the initial MST.
- Repeat the following steps until all vertices are in MST:
  - Find all the edges from the newest vertex to other vertices that are not in the tree.
  - Pick the edge with the lowest weight and add this edge and its destination vertex to the tree.
  - Update accordingly the associated weights of other vertice not on the MST currently

  - What data structure should we use to keep edges?
    - Put the edges in a priority queue.

Prim's Algorithm for finding MST

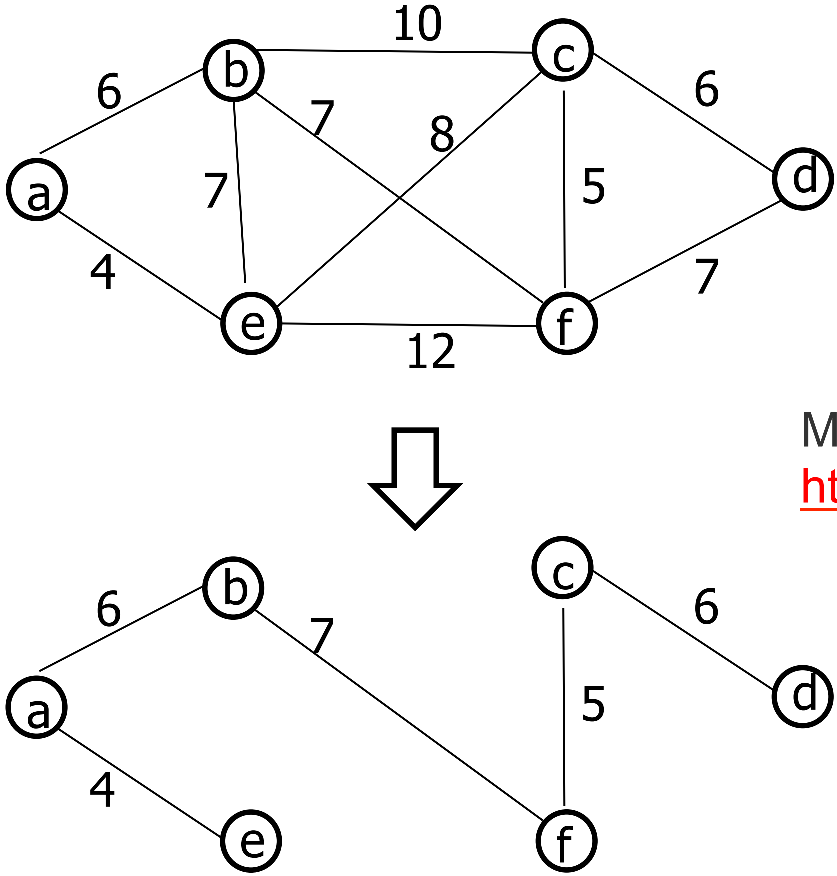
- Example:



More examples:
https://visualgo.net/en/mst

Prim's Algorithm for finding MST

- Example:



More examples:
https://visualgo.net/en/mst

Prim's Algorithm - Analysis

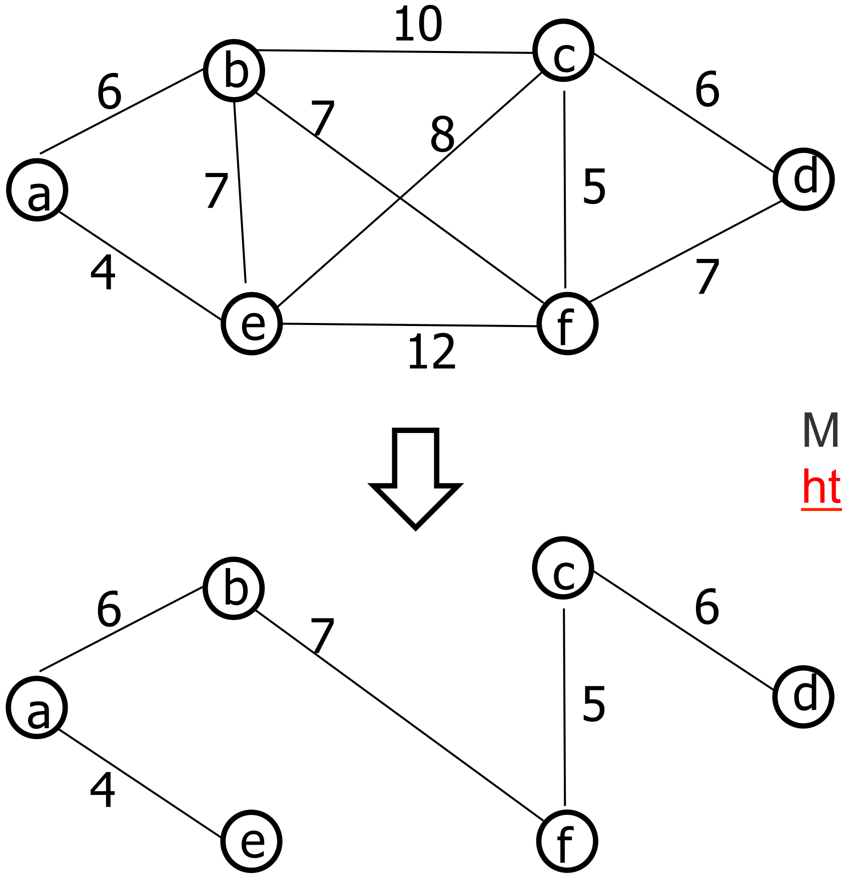
- What is the size of the priority queue?
  - O(N), where N is the number of nodes
- How many time do we extract min from priority queue?
  - O(N) times
  - Each extract min is O(log(N))
- What is the total time complexity?
  - Note that we actually end up looking at all edges and update the keys in the priority queue if needed, which takes O(E*log(N)), E being the number of edges.
  - Total time is O(N*log(N) + E*log(N))
  - We normally have V<<E, and therefore the time complexity is O(ElogN)

Kruskal's Algorithm for finding MST

- Initially, the MST contains all vertices but no edges.
- Sort the edges in a non-decreasing order.
- Go over (sorted list of) edges and add an edge to the MST.
  Important: If the edge makes a cycle, ignore it.
- Continue until N-1 edges are added, when MST is formed.
  - Recall that MST has N-1 edges

Kruskal's Algorithm for finding MST

- Example:



More examples:
https://visualgo.net/en/mst

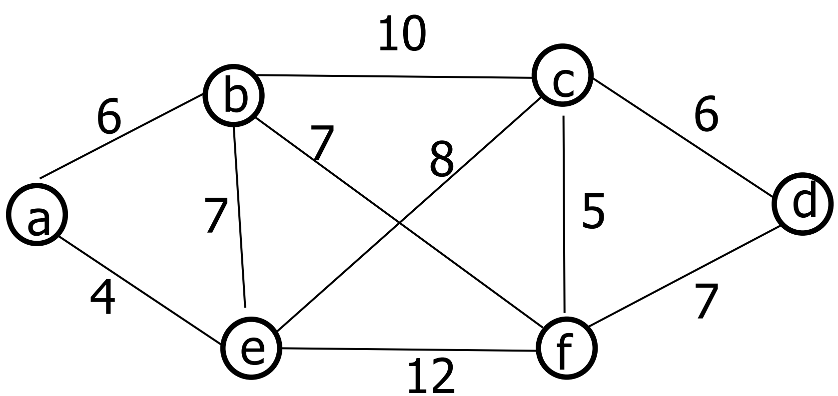
Avoiding Cycles in Kruskal’s Algorithm

- How do we know if a new edge makes cycle?
  - Group vertices, and if the edge connects two vertices of the same group, it makes a cycle.
  - How to group vertices?
    - Use a data structure called disjoint sets
      - Every set has a unique id, basic operations are: makeSet(), find() and union().
      - All the operations of disjoint sets can be implemented in O(logN)
    - First we make N sets(one for each vertex)
    - Every time we want to add an edge *ab* we check if(find(a)==find(b)), if true, it makes a cycle. Otherwise, we add the edge *ab* to MST and we union sets containing *a* and *b,* union(a,b).

Kruskal's Algorithm-Example

- Initial sets:

  {a}, {b}, {c}, {d}, {e}, {f}



Sorted list of edges:

{ae}
{cf}
{ab}
{cd}
{be}
{bf}
{fd}
{ec}
{bc}
{ef}

Kruskal's Algorithm - Analysis

- What is the time needed to sort edges?
  - O(E*log(E))
- How many possible iterations?
  - O(E) to go over the sorted list of edges
  - At each iteration we check for a cycle (each can be O(1) using amortized analysis)
- What is the total time complexity?
  - Total time is O(E*log(E) + E)